Lab3 Report: SDN Open Virtual Switches

** Please **fill in the report** and submit the **pdf** to NYU Brightspace*

Name: Yujie Yu, Junda Ai ID: yy3913, ja4426 Date: 2022/10/27

1. Objectives

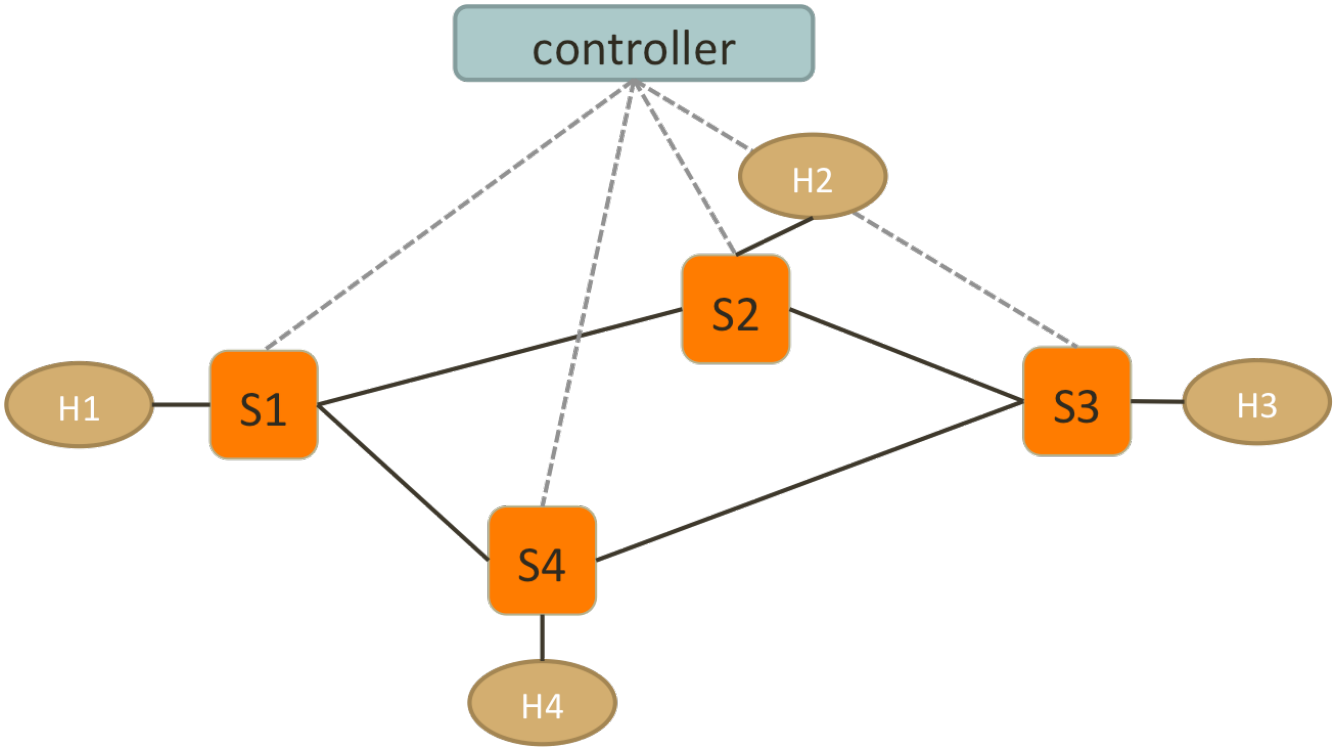
- Understand SDN and get familiar with controllers.

2. References

- https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_switch_13.py
- https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html
- Slides

3. Experiments

1. Use Mininet to create the following topology: (4 Hosts, 4 OVSes) with a remote controller
2. Use RYU to implement the controller (you can use other controller such as BEACON, POX, etc...)



3. Test Connectivity using ping. (Hint: take care of ARP packets in the controller and install proper rules for them.)
4. Enforce *these policies*:
    - **Everything follows shortest path**
    - **When there are two shortest paths with equal costs available**
        - ICMP and TCP packets take the clockwise path
            - e.g. S1-S2-S3, S2-S3-S4
        - UDP packets take the counterclockwise path
            - e.g. S1-S4-S3, S2-S1-S4
        - H2 and H4 cannot send HTTP traffic (TCP with dst_port:80)
            - New connections are dropped with a TCP RST sent back to **H2 or H4**
            - To be more specific, when the first TCP packet (SYN) arrives **S2 or S4**, forwarded it to controller, controller then create a RST packet and send it back to the host.
        - H1 and H4 cannot send UDP traffic
            - simply drop packets at switches

Important! Handle the flow rules in Packet-In and let the controller handles the rules dynamically.

If you use static rules for those policies or handle them in SwitchFeatureHandler, your lab score will be removed.

4. Reports

(a) Screenshots of your mininet with "pingall", **before** and **after starting the controller**.

```
yuji@yuji:~/ctest$ sudo mn --custom topology.py --topo mytopo --mac --switch ovsk --controller remot
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s1, s2) (s2, s3) (s3, s4) (s4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
mininet>
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

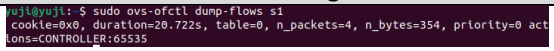
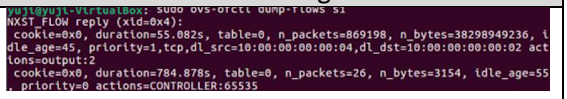
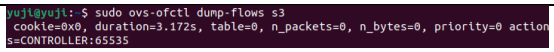
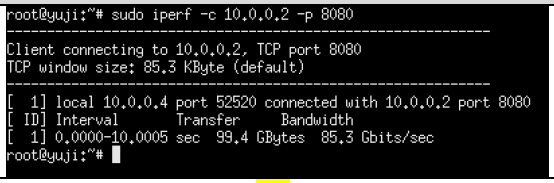
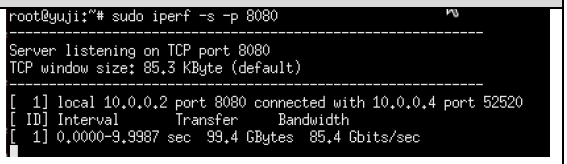
(b) How do you generate different traffic? Which tools do you use to generate: ICMP, TCP, UDP and HTTP traffic?

Hping3. With hping3 it is possible to generate TCP, UDP and HTTP traffic, and HTTP is a type of TCP traffic.

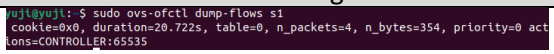
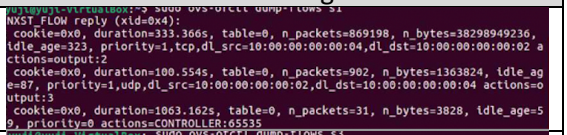
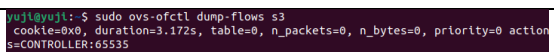
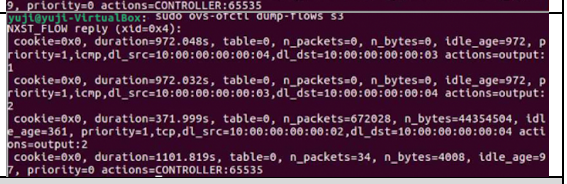
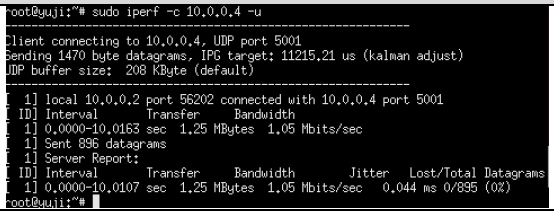
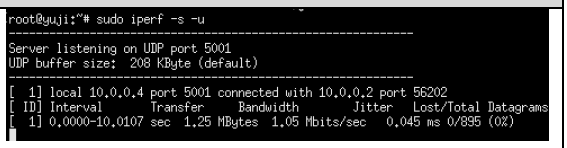
(c) Generate ICMP flows from **H4 to H3**, and take **screenshots** of the flow table on **S2** and S3 before and after the flow is generated to show that your flow follow the right path. (ovs-ofctl dump-flows)

	Before ICMP flow is generated	After ICMP flow is generated
S2	yuji@yuji:~$ sudo ovs-ofctl dump-flows s2 cookie=0x0, duration=2.617s, table=0, n_packets=0, n_bytes=0, priority=0 action s=CONTROLLER:65535	yuji@yuji-VirtualBox:~$ sudo ovs-ofctl dump-flows s2 NXST_FLOW reply (xid=0x4): cookie=0x0, duration=234.988s, table=0, n_packets=18, n_bytes=2058, idle_age=4, priority=0 actions=CONTROLLER:65535
S3	mininet> sh ovs-ofctl dump-flows s3 cookie=0x0, duration=14.434s, table=0, n_packets=0, n_bytes=0, in_port="s3-eth1 " actions=output:"s3-eth2" cookie=0x0, duration=3.429s, table=0, n_packets=0, n_bytes=0, priority=500,tcp, in_port="s3-eth1" actions=output:"s3-eth2" mininet>	yuji@yuji-VirtualBox:$ sudo ovs-ofctl dump-flows s3 NXST_FLOW reply (xid=0x4): cookie=0x0, duration=47.023s, table=0, n_packets=0, n_bytes=0, idle_age=47, pri ority=1,icmp,dl_src=10:00:00:00:00:04,dl_dst=10:00:00:00:00:03 actions=output:1 cookie=0x0, duration=47.007s, table=0, n_packets=0, n_bytes=0, idle_age=47, pri ority=1,icmp,dl_src=10:00:00:00:00:03,dl_dst=10:00:00:00:00:04 actions=output:2 cookie=0x0, duration=176.794s, table=0, n_packets=18, n_bytes=2086, idle_age=47 priority=0 actions=CONTROLLER:65535

(d) Generate TCP flows (dst_port: 8080) from **H4 to H2**, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the TCP traffic.

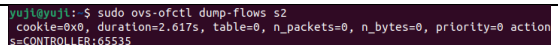
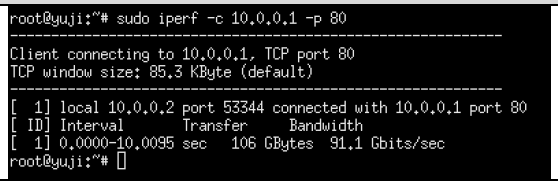
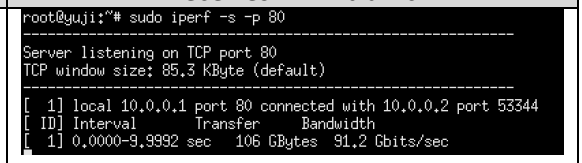
	Before TCP flow is generated	After TCP flow is generated
S1		
S3		
	Generates TCP traffic	**Receives TCP traffic**
Mininet or hosts		

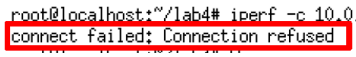
(e) Generate UDP flows from **H2 to H4**, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

	Before UDP flow is generated	After UDP flow is generated
S1		
S3		
	Generates UDP traffic	**Receives UDP traffic**
Mininet or hosts		

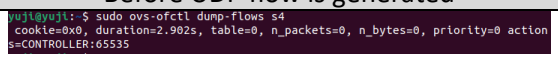
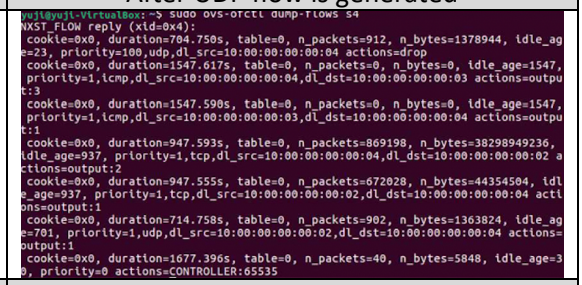
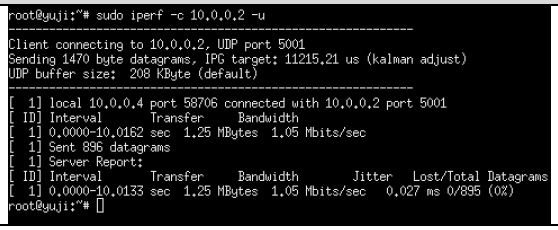
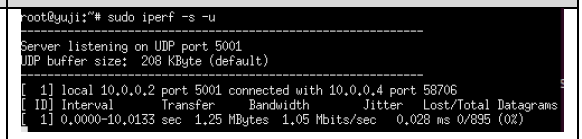
(f) Generate HTTP traffic from **H2 to H1**, and take **screenshots** of the flow table on S2 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the HTTP traffic.

	Before HTTP flow is generated	After HTTP flow is generated

S2		
	Generates HTTP traffic	Receives HTTP traffic
Mininet or hosts		

Note: "**Connection refused**" means the RST packets is successfully sent back to S2. Otherwise, you need to check if your RST packets is correct. e.g., 

(g) Generate UDP traffic from **H4 to H2**, and take **screenshots** of the flow table on S4 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

	Before UDP flow is generated	After UDP flow is generated
S4		
	Generates UDP traffic	Receives UDP traffic
Mininet or hosts		

(h) Please find what is "Spanning Tree" and "Spanning Tree Protocol"? What's the purpose of the protocol?

Spanning Tree is a type of tree that contains no loops and reaches all nodes from the root

Spanning Tree Protocol is a network protocol that creates a topology without any loops (which would result in massive issues when broadcast might happen). The STP creates a spanning tree to reach all points from a starting point while avoiding loops and redundant paths in each network.

(i) Is it necessary to implement spanning tree in SDN for packet forwarding? Why?

Yes, it is necessary to implement STP in SDN for packet forward as it allows to make sure no loops will be taken and there is way to get priorities to certain flows.

(j) If you want to find spanning tree in SDN, how will you implement and what is the difference between traditional "Spanning Tree Protocol" and the one in SDN?

In SDN we take advantage of the fact we have a lot of control when running STP. We select the root bridge, then decide the role of ports and check their port state change.

Running RYU, we can also see the role and state of each code

(k) List three advantages of using OpenVSwitch and SDN controller compared to IP networks. Briefly explain why

Advantages include guaranteed content delivery, low operating costs and a big picture management system. The first is because we are directly linking traffic switch to switch, port to port to achieve content delivery. Operating costs are low because to change an infrastructure we are simply adding and deleting flows on the table. Lastly, using the controller we can have high level control over the whole network.

(l) Include the controller's code.

(Upload with your report or attach a sharable link)
https://drive.google.com/file/d/1mTsvoop9ZGTP_fZukiFbotUlWp_ZB69d/view?usp=share_link

(m) Include the topology file

(Upload with your report or attach a sharable link)
https://drive.google.com/file/d/1mXQ_SWB9EDz-lTc1XXLcslSxmGS6dB8c/view?usp=share_link

(n) Challenges you've encountered while doing this experiment, and explain how you manage to solve them. If you do not experience any problem, simply say no problem.

The code is more than hundred lines to write. And that's recovery the realization of Internet Protocol course about ipv4 connection.

We have zero tolerance to forged or fabricated data!! A single piece of forged/fabricated data would bring the total score down to zero.